UOTTAWA

ARDUCOPTER ON CRAZYFLIE

Yuvraj Cheema

Department of Mechanical Engineering

2025

Introduction

The purpose of this document is to provide the necessary information to operate arducopter on a crazyflie. All steps in this Document were done using an M1 macbook and a crazyflie 2.1 although it should be fairly straightfoward to mimic these steps on any machine. Up to paramters will get the crazyflie working properly, the rest of the sections are not strictly required and are not finished yet. If you have any questions, feel free to email me at `yuvrajscheema@gmail.com`. More information can also be found here. This project was inspired by the blogpost Crazyflie goes Ardupilot, and we wanted to give more detailed instructions on the setup and expand on their work.

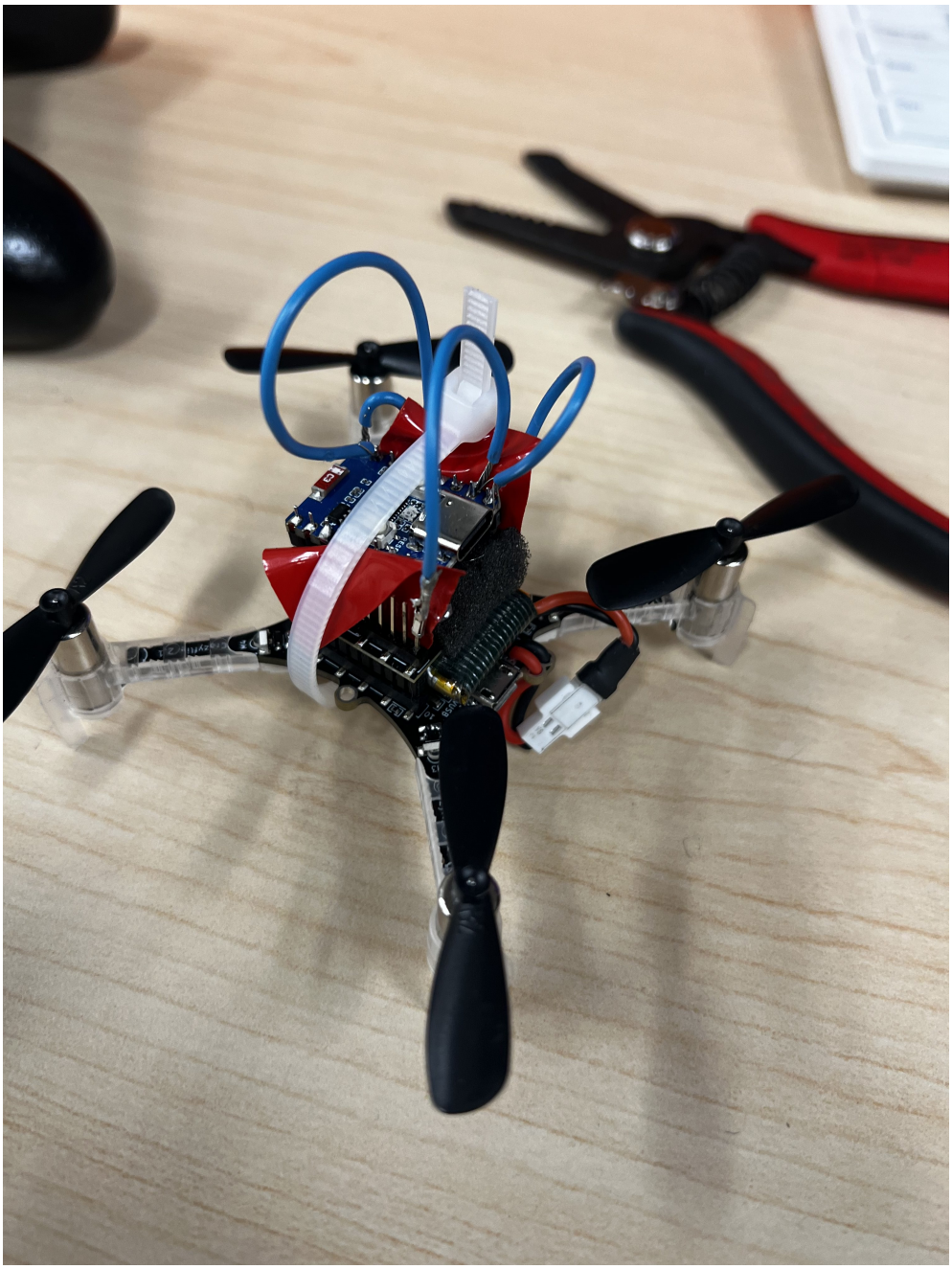

Figure 1: First version of Crazyflie with ESP32

Figure 2: Current version of Crazyflie with ESP32

Contents

Flashing Adrupilot

Downloading the Ardupilot Firmware

- Install QGroundControl, if you are using Linux or Windows you may also use MissionPlanner
- Install the STM32 CubeProgrammer
- Download the Arducopter firmware for the crazyflie here
- From that link download the arducopter with bl.hex file

Flashing the Firmware

- Disconnect the battery from the Crazyflie
- Plug connection cable into computer
- Hold down the power button
- Keep button pressed while plugging in the cable to the Crazyflie
- hold the button down for five seconds and then the Crazyflie should have a single blinking blue light and should show up in lsusb as STMicroelectronics STM32 BOOTLOADER

- Now in the the STM32CubeProgrammer
  - Select USB as the connection option
  - Choose the correct port, if no port is shown, the Crazyflie is not in DFU mode, redo the previous steps on the Crazyflie
  - Click connect
  - Open the hex file downloaded earlier
  - Click download to flash it to the drone, once the download is completed restart the crazyflie

Connecting to the Crazyflie

Wired Connection

Connection to the Crazyflie with a wire is very simple, just plug the microusb into the Crazyflie and connect to your machine. Then open QGroundControl and it should autoconnect. If that does not work, open the application settings, go to comm links and create a new comm link of type serial and select the port corresponding to the crazyflie.

Wireless Connection

We will be connecting using a ESP32-C3

- The ESP32 was bought from this page here
- To flash the required firmware to the ESP32 follow the guide found here
- Once the firmware is flashed, set the mode as Access point, the baud rate to 115200, and the serial protocol to Mavlink. Do not touch the RTS and CTS pins, for the RX and TX pins you may use any allowable pins of your choice, in this example TX was pin 4 and RX was pin 5.
- The wiring was done using the figures 3 and 4
- Use the UART2 pins on the crazyflie for the serial communication

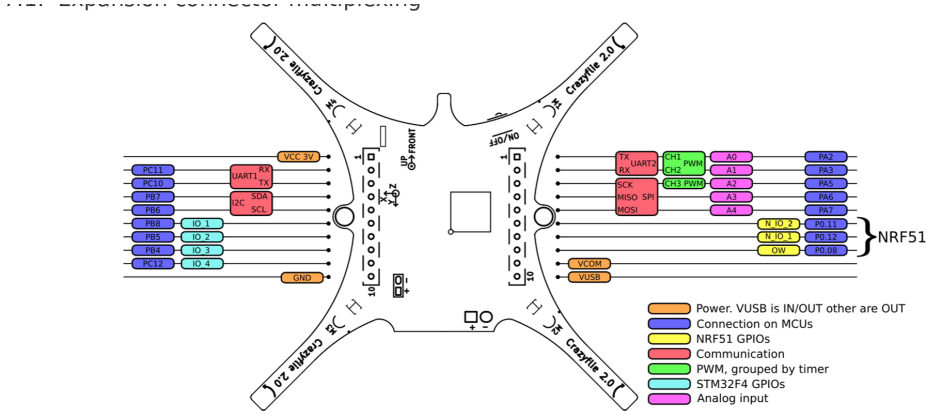

Figure 3: Crazyflie 2.1 IO

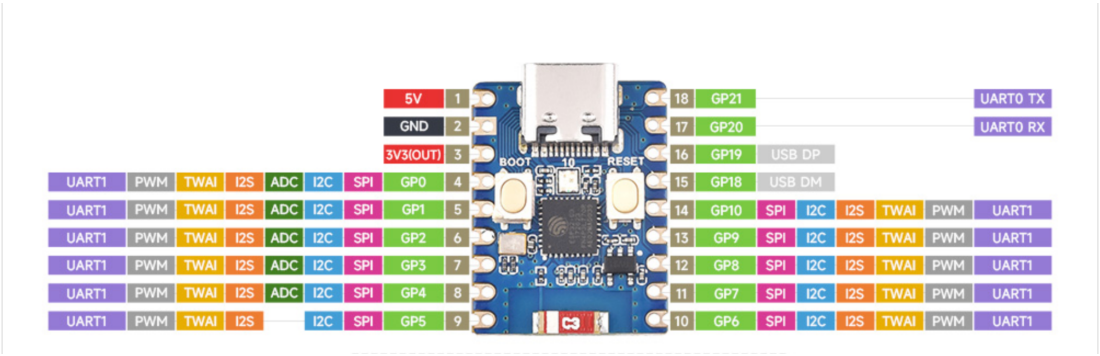

Figure 4: ESP32-C3 IO

Note the numbers in green correspond to the pins selected for TX and RX, so if TX was 4 it would be GP4 not GP0

- The connections made are:
  - VCOM on crazyflie ⟷ 5V on esp32
  - GND ⟷ GND
  - UART2 TX ⟷ RX (GP5)
  - UART2 RX ⟷ TX (GP4)

As shown in figure 5

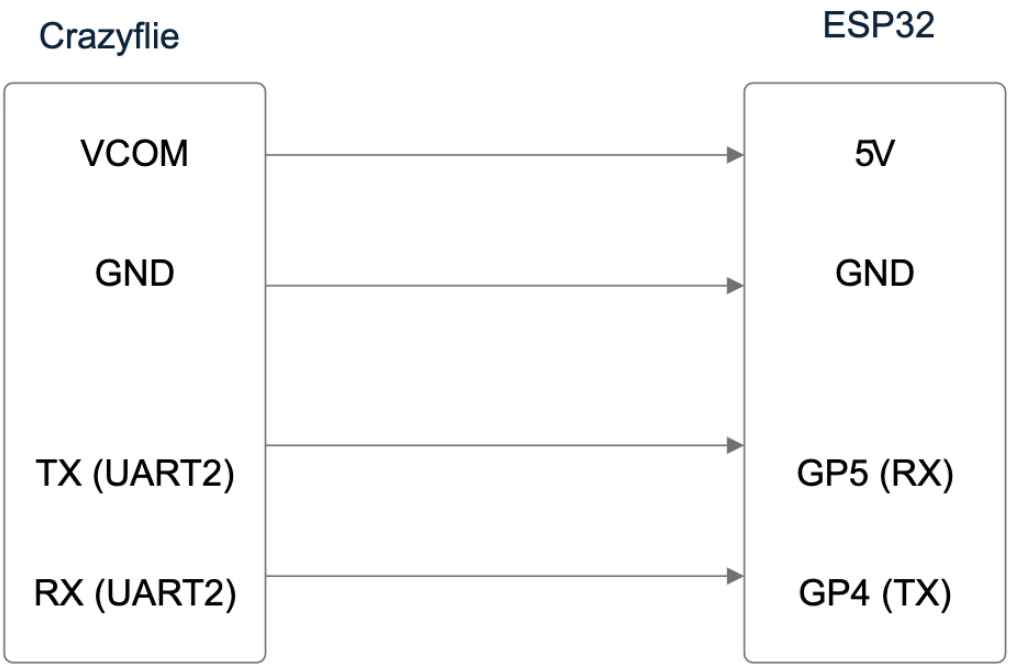

Figure 5: Connections from the Crazyflie to the ESP32

- Now connect to the Crazyflie with the wire again and go to the serial parameters
- Set every communication mode to Mavlink 2 and every baud rate to 115200
- Power the Crazyflie with a battery and connect to its wifi, QGroundControl should automatically connect to the Crazyflie, give it some time
- If that does not work, you can also try connecting via UDP or TCP
- If after doing all of this you still cannot connect to the Crazyflie, try replacing the ESP32 with a serial to USB chip and connect to the Crazyflies UART directly using a serial comm link and see if that works to isolate if the issue is with the ESP32 or the Crazyflie

Parameters

Most of the parameters were made using the ardupilot methodic configurator

- Before setting these parameters the Crazyflie would fly with a constant yaw rate which made it quite difficult to control
- The parameters can be found here
- After some testing it appears that the telemetry is attached to serial 3 since changing the baud rate of serial 3 makes it so we cannot connect to the esp32 anymore
- Almost all the parameters were either calculated by the methodic configurator, set to default or taken from the diatone mxc taycan template
- In these parameters, the slew rate was arbitrarily set to 25. Change it to zero if it seems to be causing negative effects

- Some parameters that may be worth looking further into are:
  - SCHED_LOOP_RATE
  - MOT_SPIN_ARM
  - MOT_SPIN_MIN
  - MOT_THST_EXPO
  - All the parameters that involve the notch filter
- Calibrate the acceleration and gyro after uploading these parameters, as every Crazyflie is going to be slightly different
- Now the Crazyflie should control pretty well
- Make sure to disable the geofence and pre-arm checks otherwise the Crazyflie won't arm

Connecting with the CrazyRadio PA

This section is not complete. The CrazyRadio PA and the nrf receiver on the Crazyflie need new drivers that require a development board from bitcraze. Although it is now known that the CrazyRadio can still communicate with the nrf chip, but they do not send Mavlink packets.
To confirm that the radio sees the Crazyflie do the following:

- First install libusb

  ```
  brew install libusb
  ```

- Next clone the Crazyflie python library

  ```
  git clone https://github.com/bitcraze/crazyflie-lib-python
  ```

- Now create a new python environment, here conda was used. On macOS, python 3.13 is not supported for this application however it should be fine for other platforms

  ```
  conda create --name crvenv python=3.12
  ```

- Now install the dependencies

  ```
  cd crazyflie-lib-python
  pip install -e .
  ```

- After a successful installation navigate to the example directory
- Turn on the Crazyflie and plug in the radio
- Now run the python script to connect to the Crazyflie

```
cd examples
python cfbridge.py
```

- The terminal should now print: Connected to ...
- This confirms that the radio can still connect to the Crazyflie
- Now custom drivers are needed for the communication with the flight controller which are still a work in progress

Using the Bitcraze Flowdeck

The goal here is to use Bitcraze's flowdeck v2 to have localization for the Crazyflie. Custom drivers were made for the flowdeck, contact me for more information, and after flashing them they appear to work inconsistently.

- The flowdeck consists of a optical flow sensor and a vl53l0x/vl53l1x lidar range sensor for the height and often times one of these sensors will not initialize properly requiring a full reboot of the Crazyflie
- Currently, a more robust initialization is being worked to attempt to resolve this issue
- When the sensors are initialized correctly, the Z Ranger works right out of the box but the optical flow sensor still needs tuning
- Once the initialization and tuning have been complete the flowdeck should work as a localization tool for the Crazyflies running Ardupilot

L1 on a Crazyflie

The goal here was to use L1Quad on a Crazyflie. The firmware was flashed however, the current lack of positioning stopped any test flights from occurring. Once the flowdeck has been fully ported, it should not be too difficult to test L1Quad. There was also an attempt made at turning the L1Quad firmware to be able to be joystick controlled but due to limitations in time this was not complete. For more information please contact the email in the introduction.

- First clone the repo into a folder add and the submodules by executing the following

```
git clone https://github.com/sigma-pi/L1Quad
cd L1Quad
git submodule update --init --recursive
```

- Next, download the requirements for this project

```
conda create --name arducopter python=3.8
conde activate arducopter
xcode-select --install
```

- We want to be using the latest version of g++ so make sure you are not using the px4 package by running and install any other arm-none-eabi installs

```
brew unistall gcc-arm-none-eabi
```

- Now install some packages

```
brew update
brew upgrade
brew install genromfs
brew install --cask gcc-arm-embedded
brew install gawk
pip install pyserial
pip install future empy
pip install pyexpect
```

- Next, implement the L1 controller

```
rm ./ardupilot/ArduCopter/Copter.h ./ardupilot/ArduCopter/Paramet
```

```
cp ./L1AC_customization/ArduCopter/ACRL_trajectories.cpp ./L1AC_c
```

- Now running the following command with the venv active should provide a list of possible boards for the firmware
- Go to the file **config.h** and set REAL_OR_SITL to 1
- Run the following to create the firmware file

```
./waf distclean
./waf configure --board crazyflie2
./waf copter
```

- That should compile properly
- Now, look in ardupilot/build/crazyflie2/bin there will be a .apj file
- To flash the new firmware, open QGroundControl and connect to the Crazyflie already flashed with Ardupilot via USB
- Next, go to the firmware tab, then unplug and replug the Crazyflie with the battery disconnected
- Now, a pop-up menu should appear

- Select Ardupilot, Chibios, Multirotor, check the advanced settings box and choose custom firmware file, then press ok
- Now select the .apj file and flash the Crazyflie with Ardupilot
- Now upload the parameters for version 4.4.4 and you can now use L1
- To actually use L1, use mavproxy, connect to the Crazyflie via USB and run

  ```
  ls /dev/tty.*
  ```

- Now you the port for the Crazyflie should appear, use that in the next command

  ```
  mavproxy.py --mavproxy.py --master=/dev/tty.****** --baudrate 115
  ```

- This may take a bit to run, but once it has, the following text will pop-up on the screen, hit enter, and the terminal will output STABILIZE¿
- Now, run the following to enter L1 mode

  ```
  mode 29
  ```